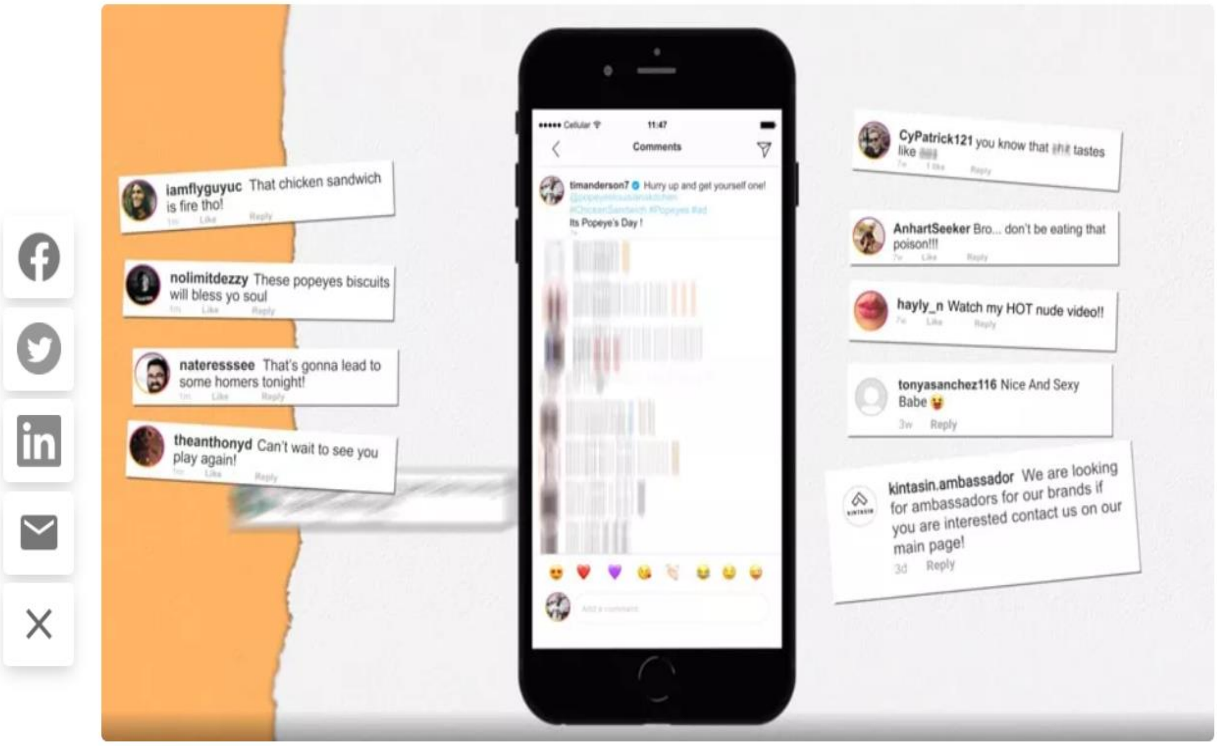
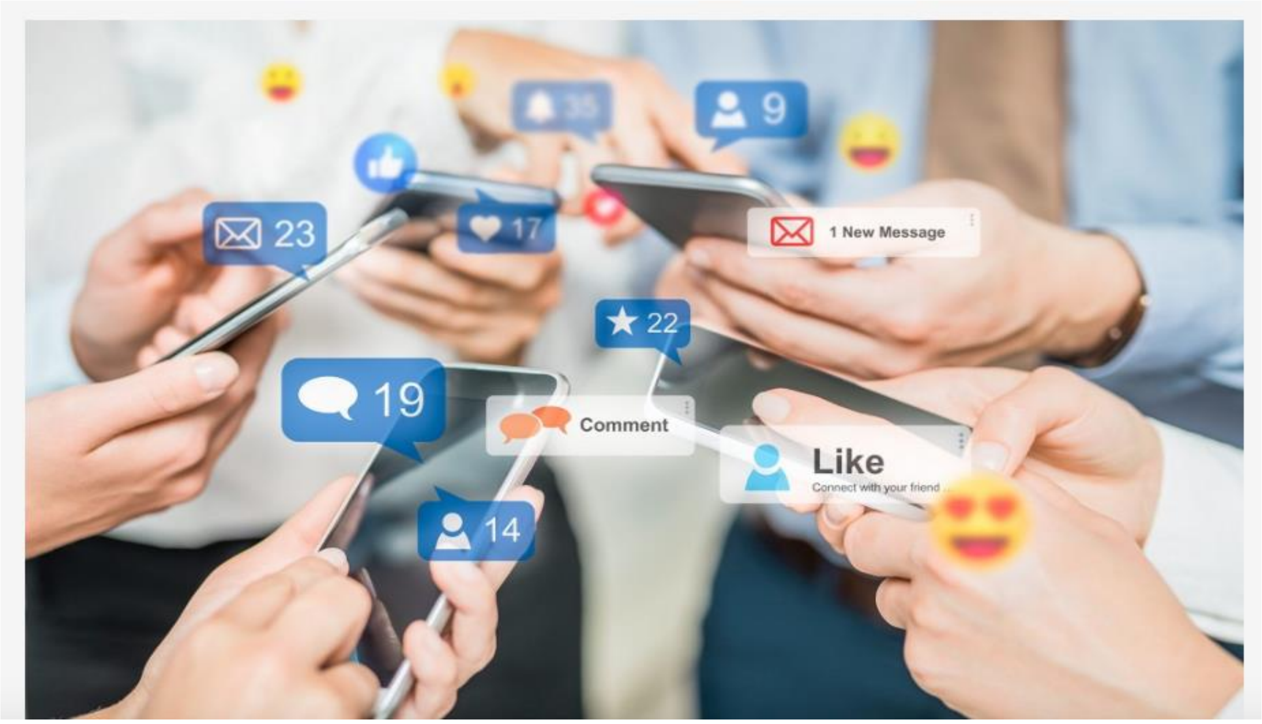
FLIP ROBO

Malignant comment Classifier Project


iamflyguyuc That chicken sandwich is fire tho!
nolimitdezzy These popeyes biscuits will bless yo soul
nateresssee That's gonna lead to some homers tonight!
theanthonyd Can't wait to see you play again!
Comments
timanderson7 Hurry up and get yourself one!
Its Popeye's Day !
CyPatrick121 you know that tastes like
AnhartSeeker Bro... don't be eating that poison!!!
hayly_n Watch my HOT nude video!!
tonyasanchez116 Nice And Sexy Babe
kintasin.ambassador We are looking for ambassadors for our brands if you are interested contact us on our main page!
Reply



Submitted by:

SARMISHTHA HALDAR

ACKNOWLEDGEMENT:

I would like to thank all my teachers, supervisors for the learning especially Shubham Yadav Sir. Few journals referred in the case are as follows:

i) Yin *Dawei ,et al Detection of Harassment on web 2.0*

ii) Liang-Chih Yul Jin Refining word embeddings for Sentiment Analysis.

Business Problem Framing

This is a classic problem in today's world of social media platform where our goal is to build a prototype which helps to evaluate the hate and abusive comments made on social media platform so that it can be controlled and restricted from spreading

Conceptual Background

Online platforms when used by normal people can only be comfortably used by them only when they feel that they can express themselves freely and without any reluctance. If they come across any kind of a malignant or toxic type of a reply which can also be a threat or an insult or any kind of harassment which makes them uncomfortable, they might defer to use the social media platform in future. Thus, it becomes extremely essential for any organization or community to have an automated system which can efficiently identify and keep a track of all such comments and thus take any respective action for it, such as reporting or blocking the same to prevent any such kind of issues in the future. This is a huge concern as in this world, there are 7.7 billion people, and, out of these 7.7 billion, more than 3.5 billion people use some or the other form of online social media. Which means that every one-in-three people uses social media platform. This problem thus can be eliminated as it falls under the category of Natural Language Processing. In this, we try to recognize the intention of the speaker by building a model that's capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate. Moreover, it is crucial to handle any such kind of nuisance, to make a more user-friendly experience, only after which people can actually enjoy in participating in discussions with regard to online conversation

INTRODUCTION

Internet comments are bastions f hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. Here our goal is to build a prototype of online hate and abuse comment classifier which can be used to classify hate and offensive comments so that it can be controlled.

Problem Statement

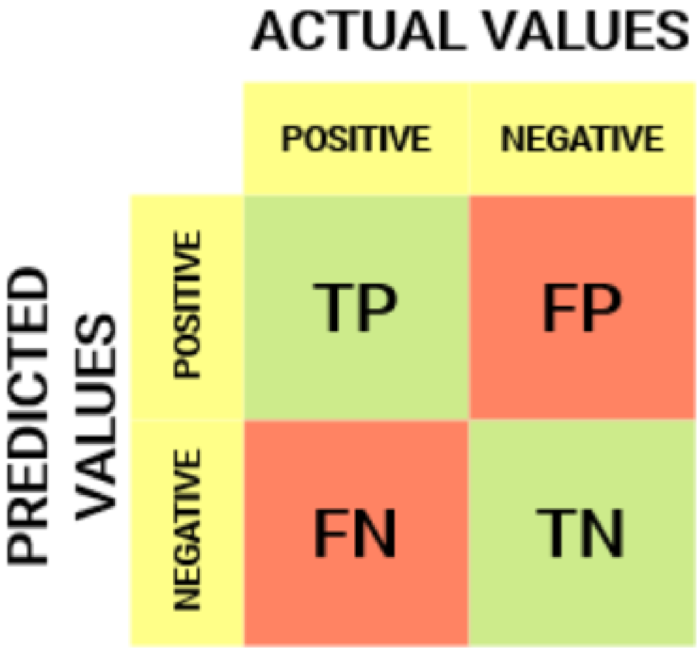
To Build a model which can be used to classify hate and offensive comments.

Analytical Problem Framing

We have used methods like Accuracy, Confusion matrix and Roc-auc curve for model evaluations

Accuracy is defined as the percentage of correct predictions

A **Confusion matrix** is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.



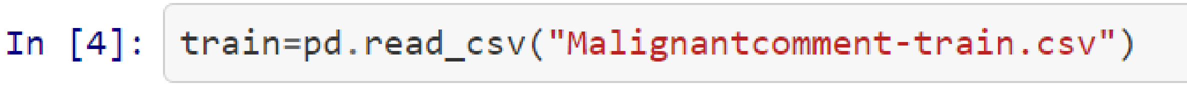
Let’s decipher the matrix:

- The target variable has two values: **Positive** or **Negative**
- The **columns** represent the **actual values** of the target variable
- The **rows** represent the **predicted values** of the target variable

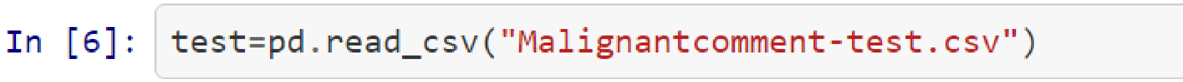
ROC Curve :The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems

- Data Sources and their formats

We received the data in the form of .csv file and data was loaded using Pandas

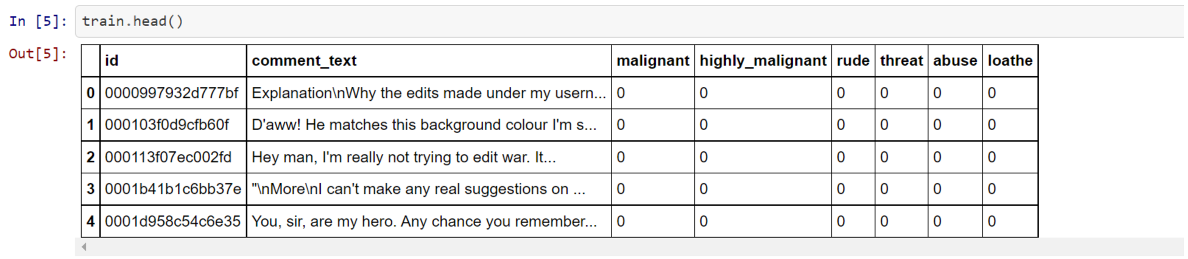


```
In [4]: train=pd.read_csv("Malignantcomment-train.csv")
```



```
In [6]: test=pd.read_csv("Malignantcomment-test.csv")
```

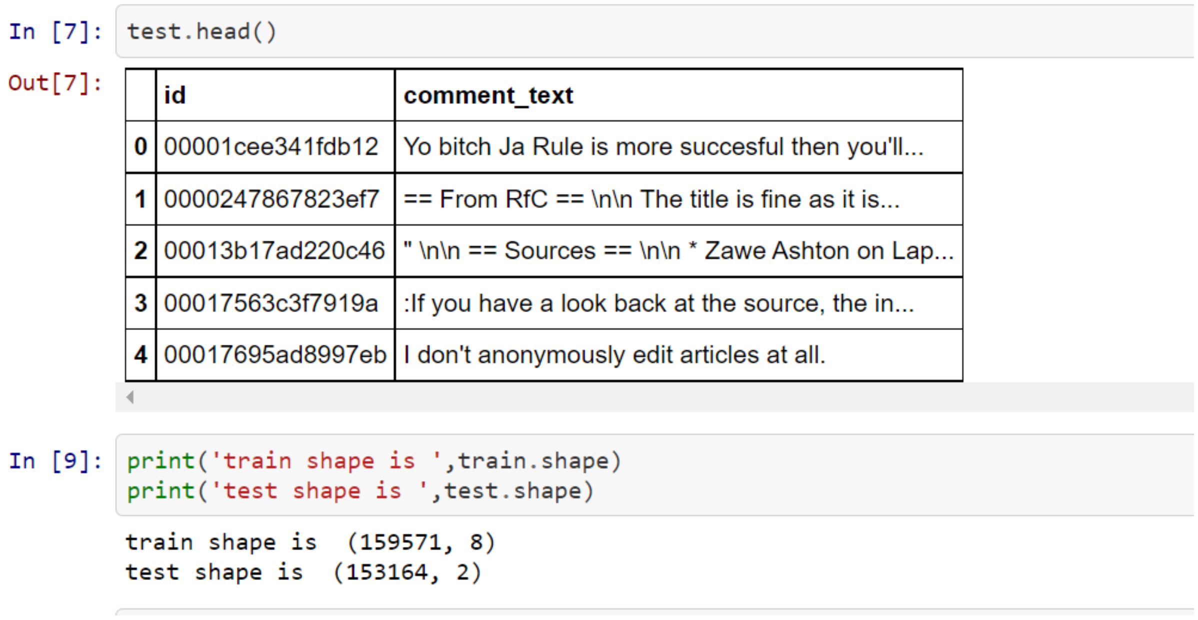
- Exploratory Data Analysis



In [5]:
```
train.head()
```

Out[5]:

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0



In [7]:
```
test.head()
```

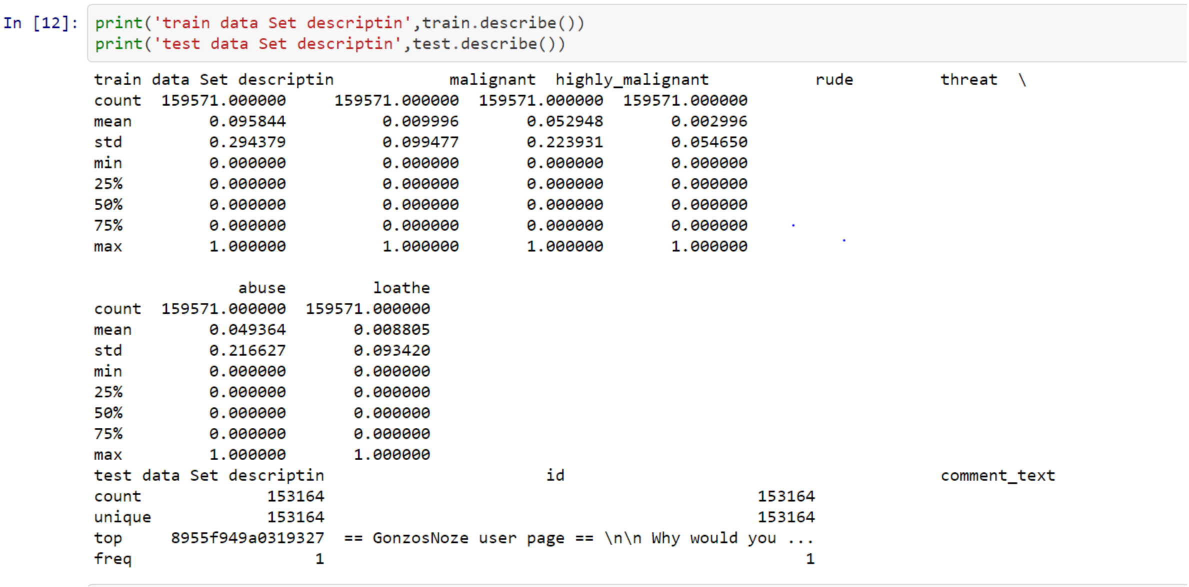
Out[7]:

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.

In [9]:
```
print('train shape is ',train.shape)
print('test shape is ',test.shape)
```

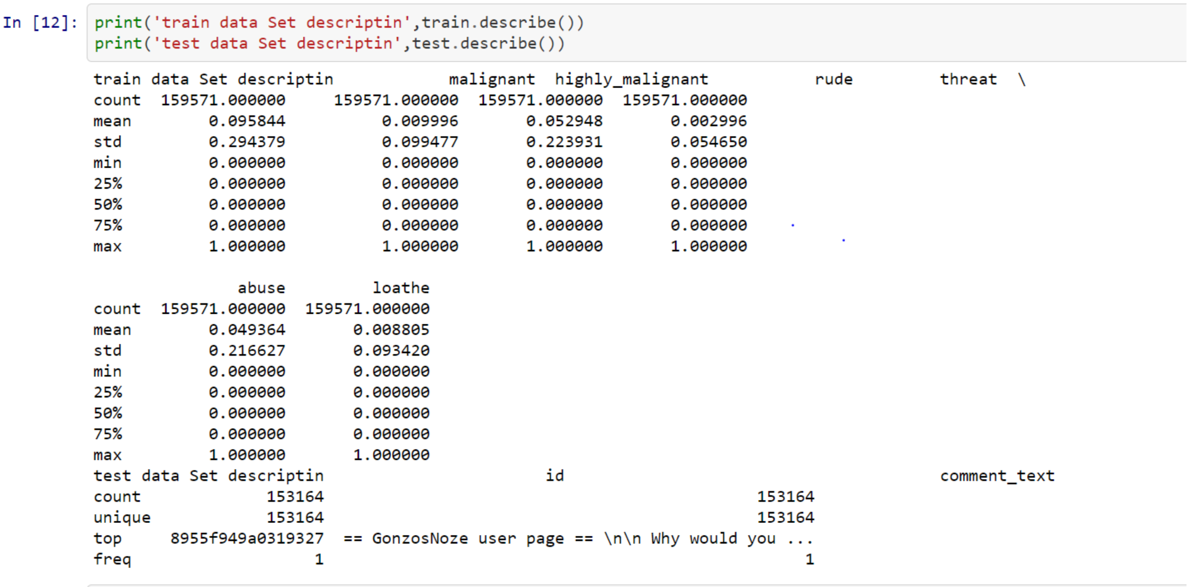
```
train shape is  (159571, 8)
test shape is  (153164, 2)
```

Descriptive Statistics



```
In [12]: print('train data Set descriptin',train.describe())
         print('test data Set descriptin',test.describe())
```

```
train data Set descriptin            malignant  highly_malignant          rude         threat  \
count  159571.000000  159571.000000  159571.000000  159571.000000
mean        0.095844       0.009996       0.052948       0.002996
std         0.294379       0.099477       0.223931       0.054650
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         1.000000       1.000000       1.000000       1.000000

               abuse         loathe
count  159571.000000  159571.000000
mean        0.049364       0.008805
std         0.216627       0.093420
min         0.000000       0.000000
25%         0.000000       0.000000
50%         0.000000       0.000000
75%         0.000000       0.000000
max         1.000000       1.000000
test data Set descriptin                       id                                       comment_text
count               153164                                              153164
unique              153164                                              153164
top     8955f949a0319327  == GonzosNoze user page == \n\n Why would you ...
freq                     1                                                   1
```

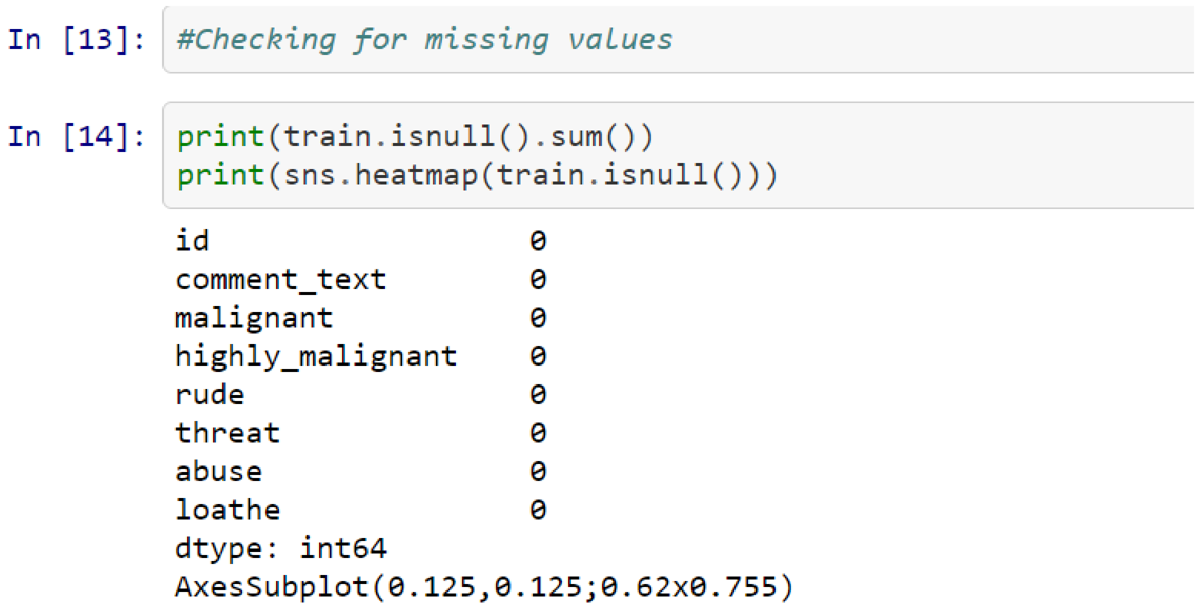


```
In [12]: print('train data Set descriptin',train.describe())
         print('test data Set descriptin',test.describe())
```

```
train data Set descriptin            malignant  highly_malignant          rude         threat  \
count  159571.000000  159571.000000  159571.000000  159571.000000
mean        0.095844       0.009996       0.052948       0.002996
std         0.294379       0.099477       0.223931       0.054650
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         1.000000       1.000000       1.000000       1.000000

               abuse         loathe
count  159571.000000  159571.000000
mean        0.049364       0.008805
std         0.216627       0.093420
min         0.000000       0.000000
25%         0.000000       0.000000
50%         0.000000       0.000000
75%         0.000000       0.000000
max         1.000000       1.000000
test data Set descriptin                       id                                       comment_text
count               153164                                              153164
unique              153164                                              153164
top     8955f949a0319327  == GonzosNoze user page == \n\n Why would you ...
freq                     1                                                   1
```

- ## Data Preprocessing

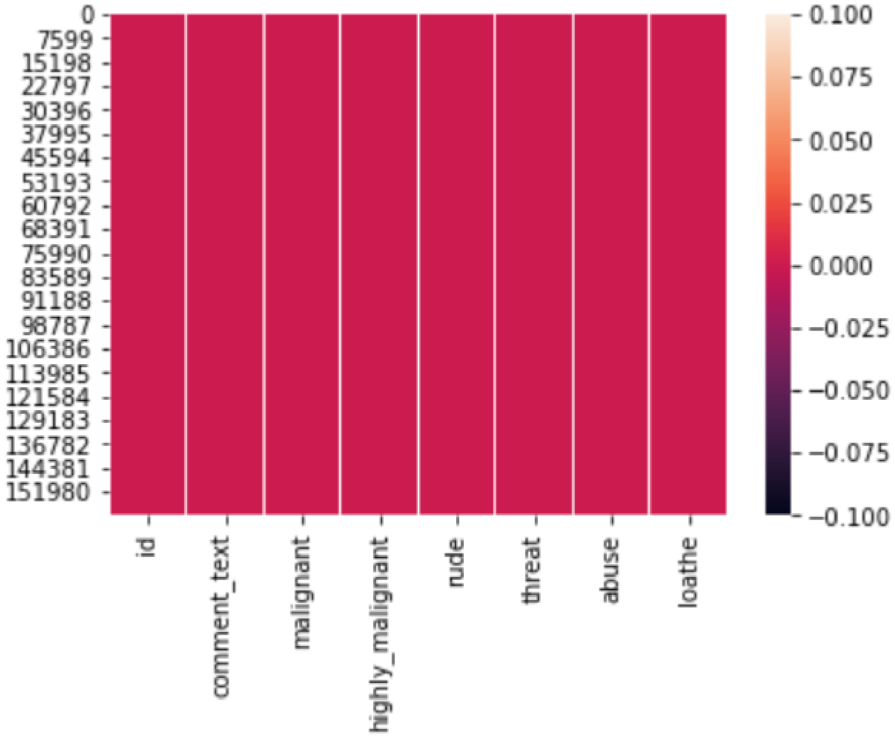
i)Checking the missing values



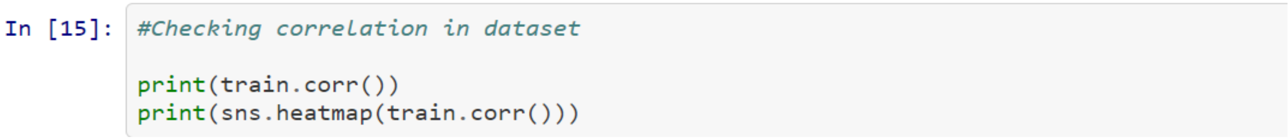
```
In [13]: #Checking for missing values
```

```
In [14]: print(train.isnull().sum())
         print(sns.heatmap(train.isnull()))
```

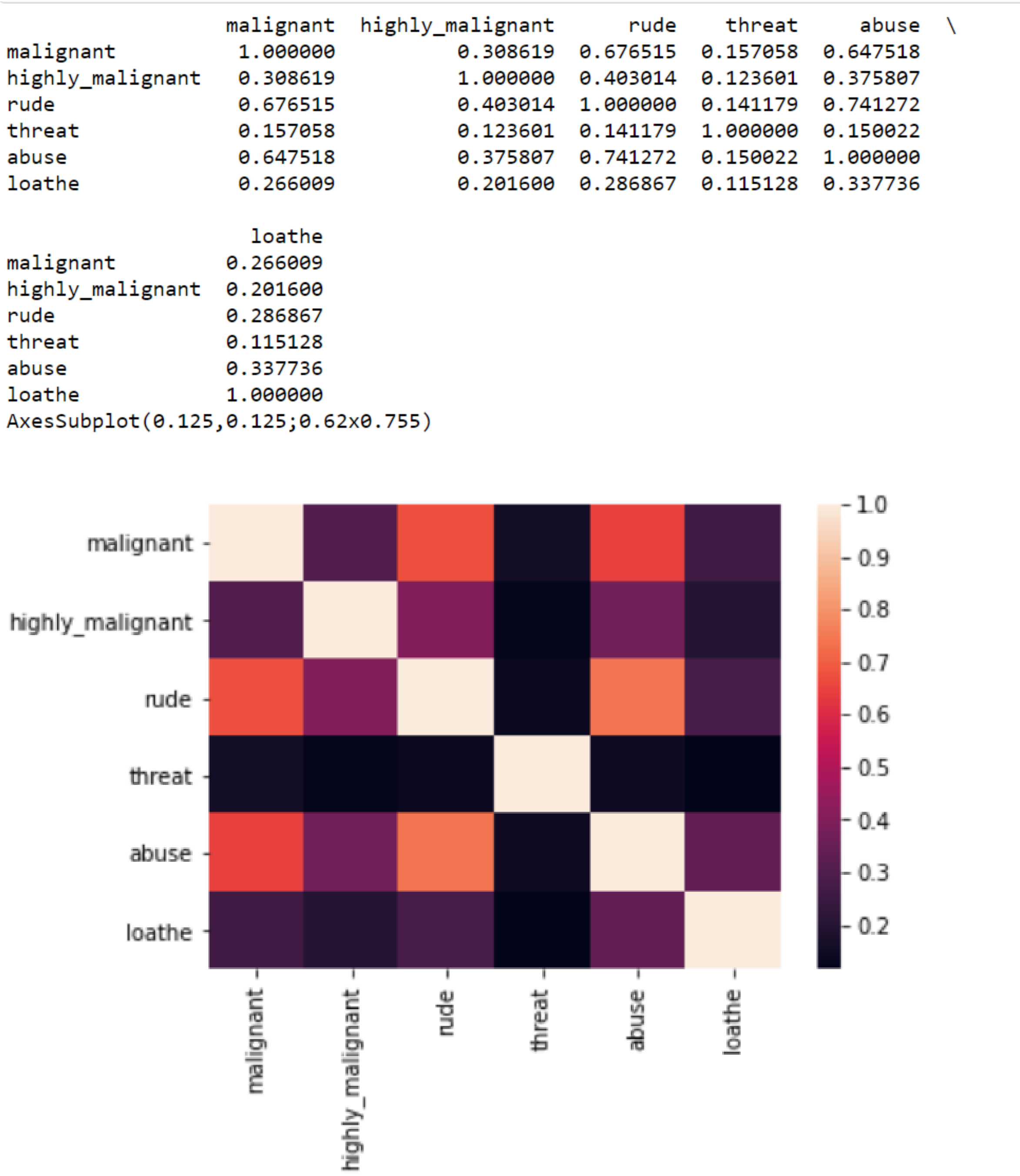
```
id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
AxesSubplot(0.125,0.125;0.62x0.755)
```



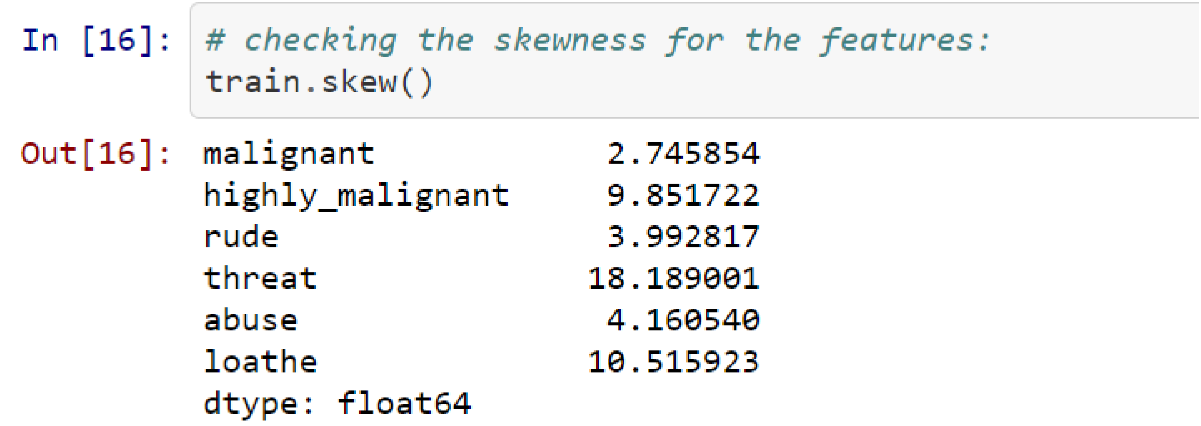
ii) Checking Correlation



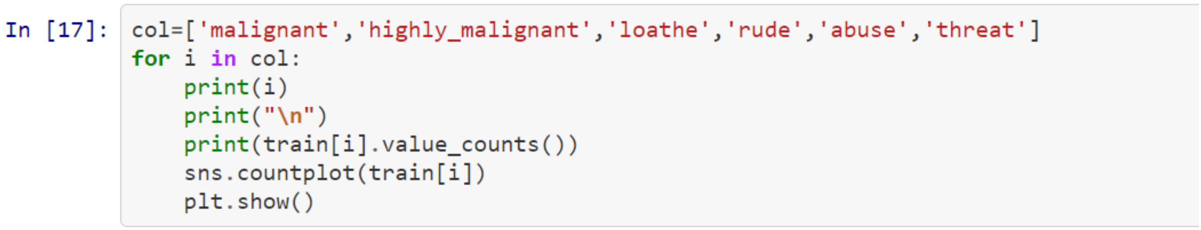
```
In [15]: #Checking correlation in dataset

         print(train.corr())
         print(sns.heatmap(train.corr()))
```



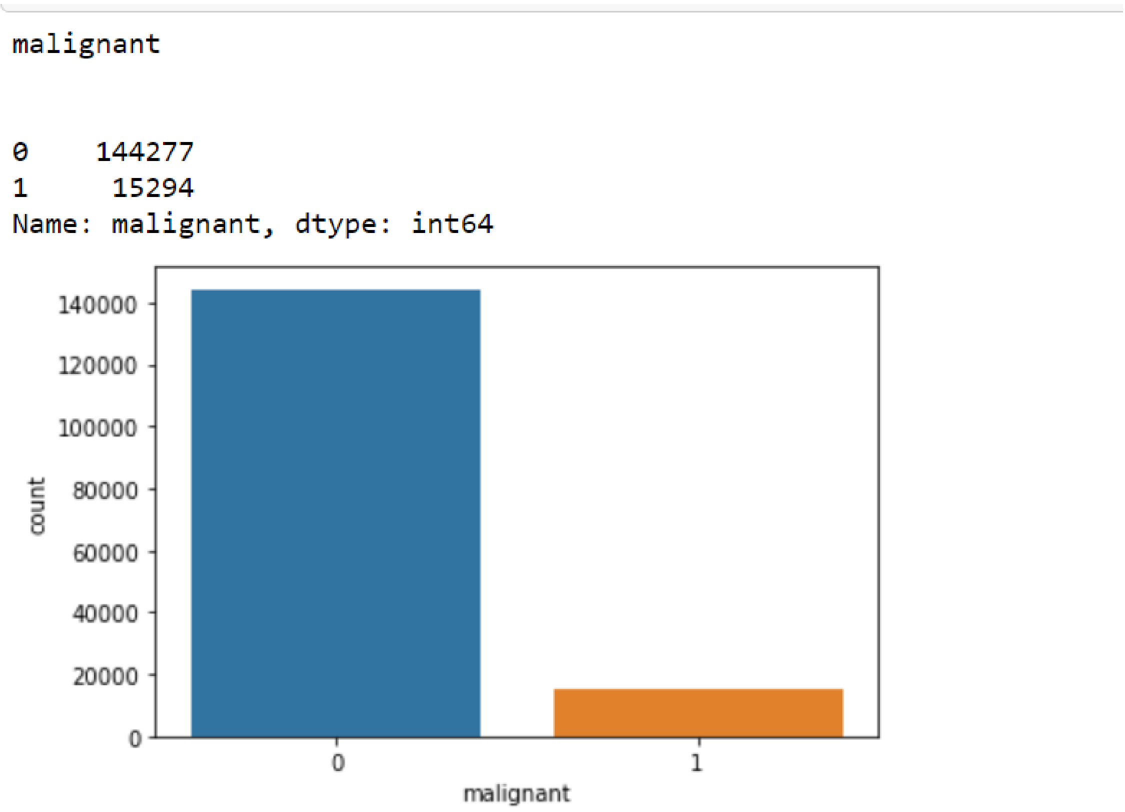
```
                   malignant  highly_malignant      rude    threat     abuse  \
malignant           1.000000          0.308619  0.676515  0.157058  0.647518
highly_malignant    0.308619          1.000000  0.403014  0.123601  0.375807
rude                0.676515          0.403014  1.000000  0.141179  0.741272
threat              0.157058          0.123601  0.141179  1.000000  0.150022
abuse               0.647518          0.375807  0.741272  0.150022  1.000000
loathe              0.266009          0.201600  0.286867  0.115128  0.337736

                    loathe
malignant         0.266009
highly_malignant  0.201600
rude              0.286867
threat            0.115128
abuse             0.337736
loathe            1.000000
AxesSubplot(0.125,0.125;0.62x0.755)
```



```
In [16]: # checking the skewness for the features:
         train.skew()

Out[16]: malignant            2.745854
         highly_malignant     9.851722
         rude                 3.992817
         threat              18.189001
         abuse                4.160540
         loathe              10.515923
         dtype: float64
```

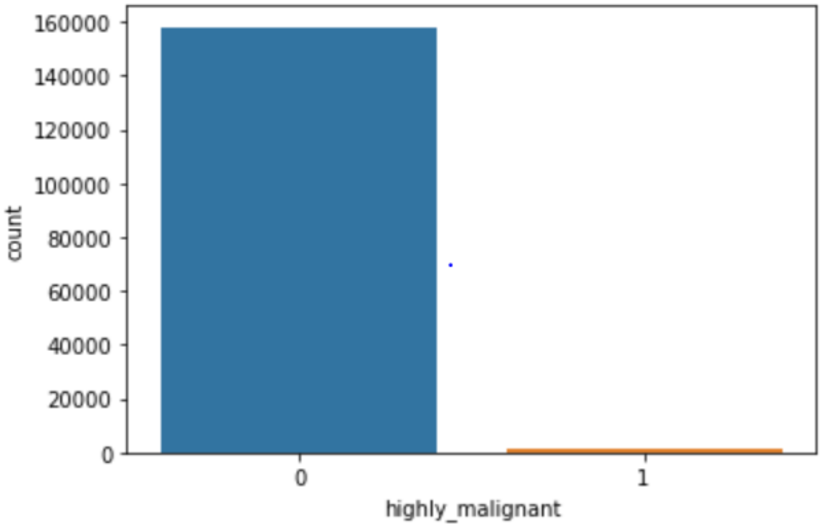


```
In [17]: col=['malignant','highly_malignant','loathe','rude','abuse','threat']
         for i in col:
             print(i)
             print("\n")
             print(train[i].value_counts())
             sns.countplot(train[i])
             plt.show()
```

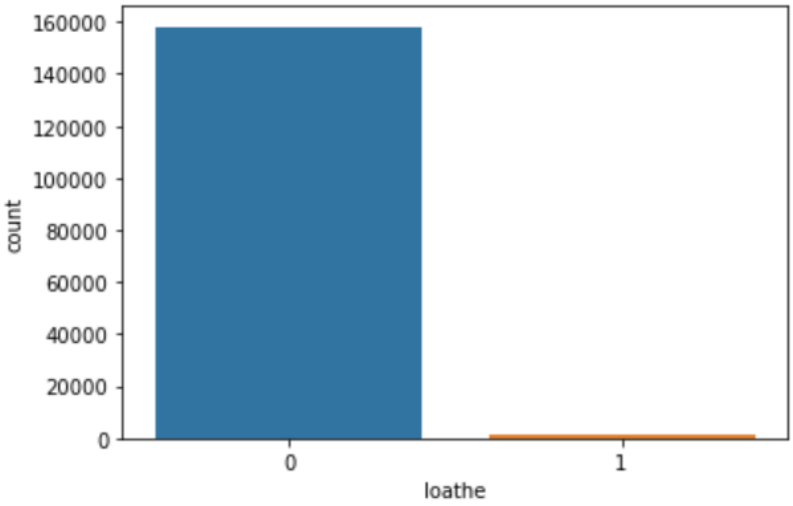


```
malignant


0    144277
1     15294
Name: malignant, dtype: int64
```

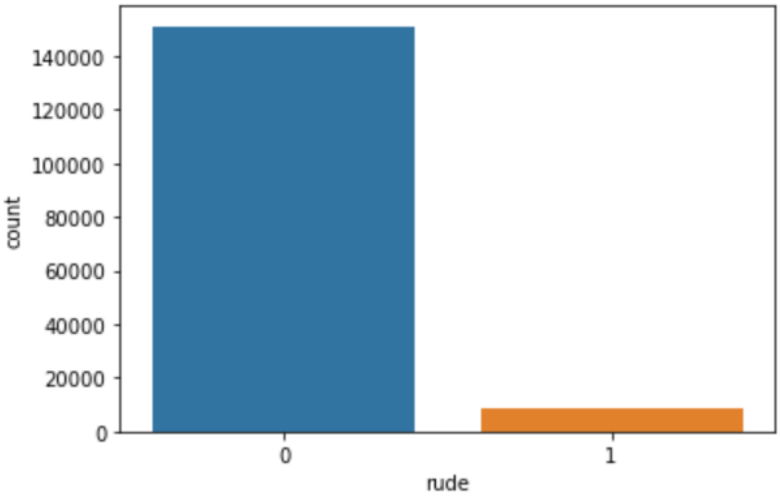
```
highly_malignant


0    157976
1      1595
Name: highly_malignant, dtype: int64
```



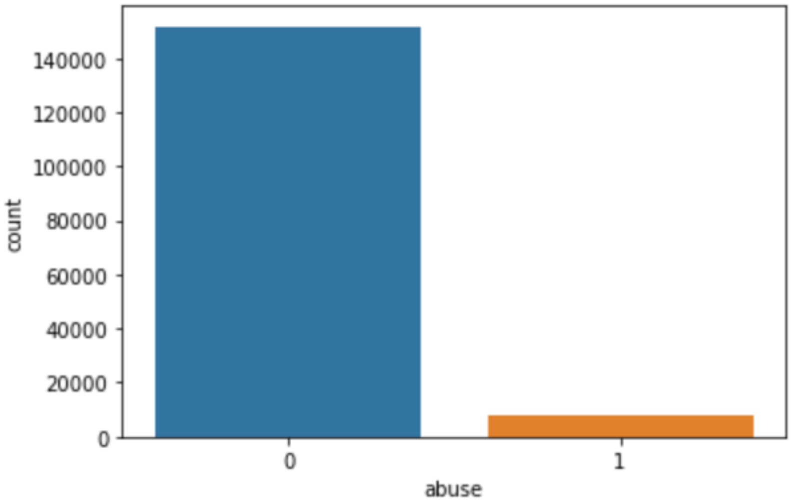
```
loathe


0    158166
1      1405
Name: loathe, dtype: int64
```

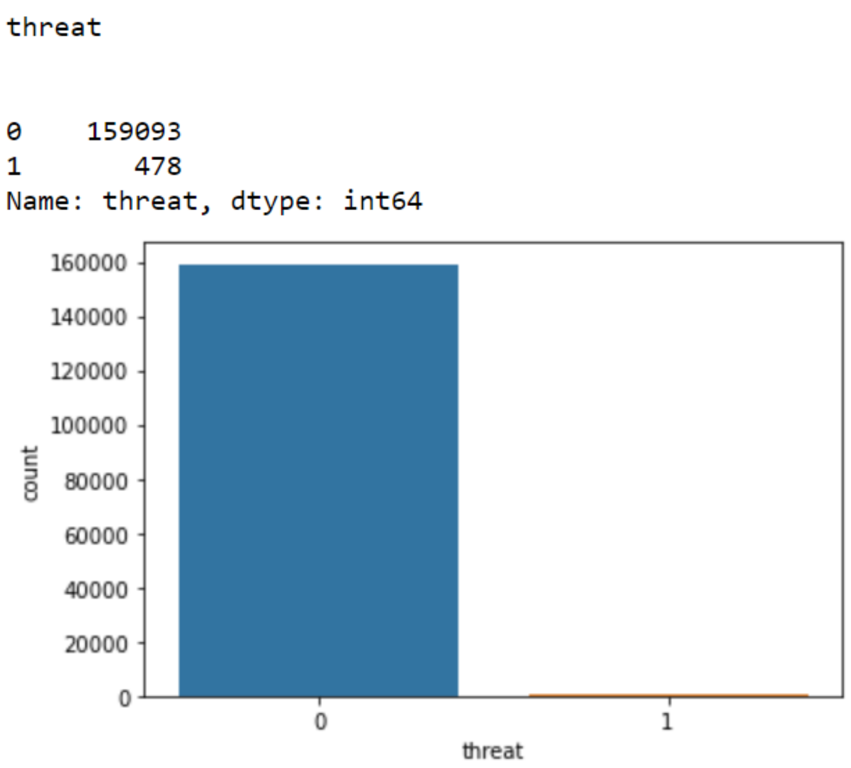


```
rude


0    151122
1      8449
Name: rude, dtype: int64
```



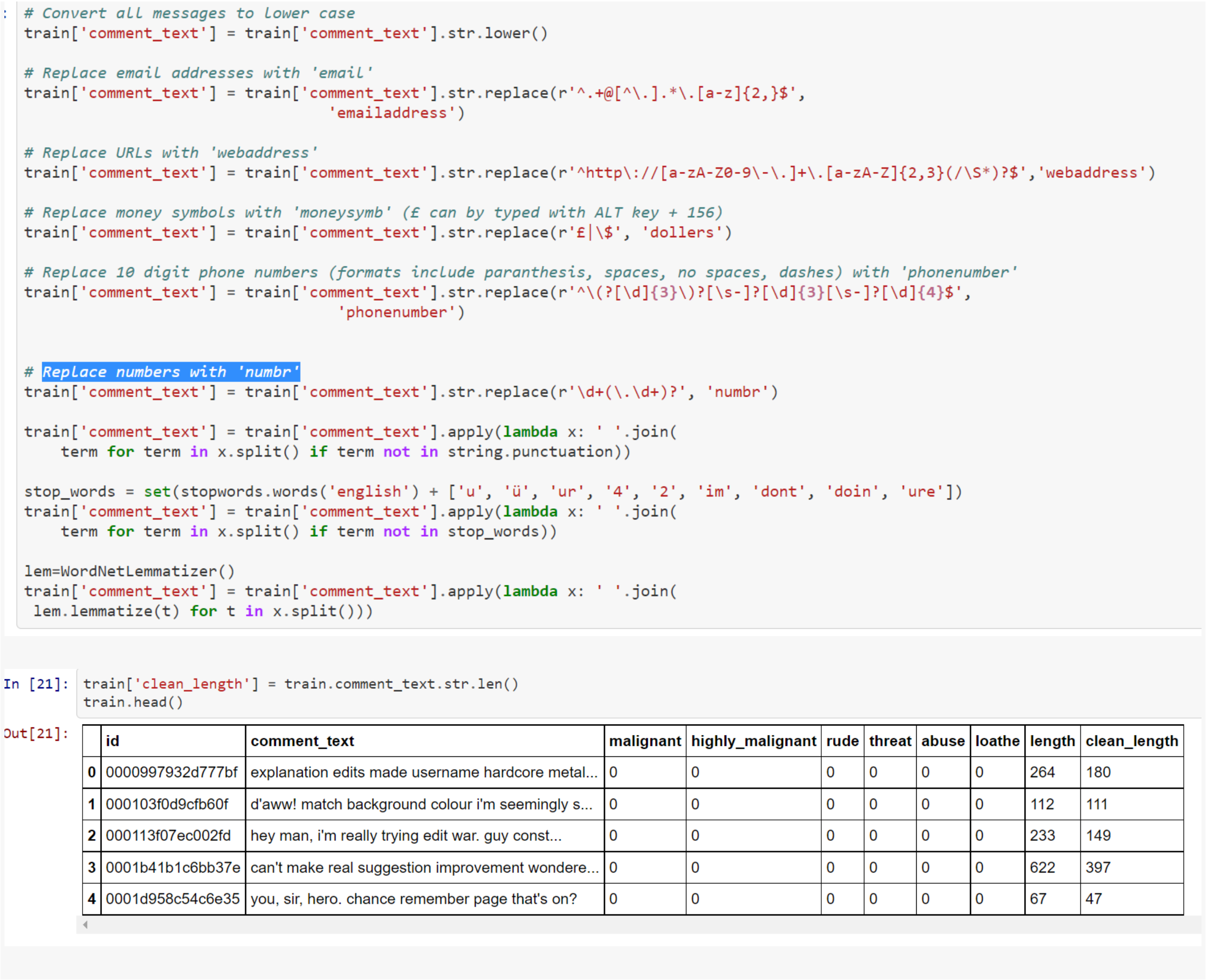
```
abuse

abuse


0    151694
1      7877
Name: abuse, dtype: int64
```





```
threat


0    159093
1       478
Name: threat, dtype: int64
```

```
iii) NLP Processing steps carried out to
convert messages to lower case,
replace email addresses with email,
replace URL’s with webaddress,
Replace 10 digit phone numbers (formats include paranthesis, spaces, no spa
ces, dashes) with 'phonenumber',
Replace numbers with 'numbr',
Stop words
```

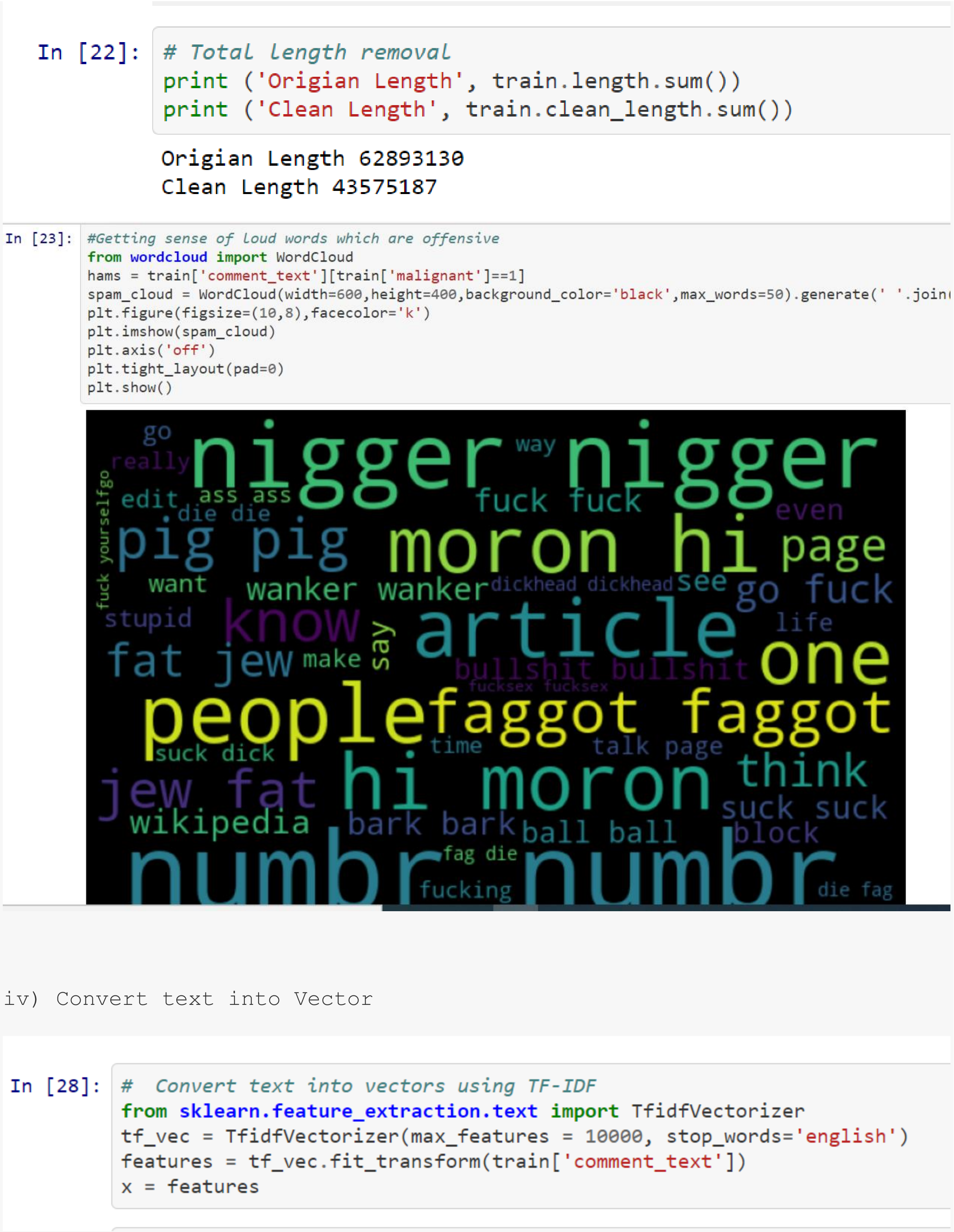


```
: # Convert all messages to lower case
  train['comment_text'] = train['comment_text'].str.lower()

  # Replace email addresses with 'email'
  train['comment_text'] = train['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                   'emailaddress')

  # Replace URLs with 'webaddress'
  train['comment_text'] = train['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','webaddress')

  # Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
  train['comment_text'] = train['comment_text'].str.replace(r'£|\$', 'dollers')

  # Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
  train['comment_text'] = train['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',
                                    'phonenumber')


  # Replace numbers with 'numbr'
  train['comment_text'] = train['comment_text'].str.replace(r'\d+(\.\d+)?', 'numbr')

  train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
      term for term in x.split() if term not in string.punctuation))

  stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
  train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
      term for term in x.split() if term not in stop_words))

  lem=WordNetLemmatizer()
  train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
   lem.lemmatize(t) for t in x.split()))
```

```
In [21]: train['clean_length'] = train.comment_text.str.len()
         train.head()
```

Out[21]:

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length	clean_length
0	0000997932d777bf	explanation edits made username hardcore metal...	0	0	0	0	0	0	264	180
1	000103f0d9cfb60f	d'aww! match background colour i'm seemingly s...	0	0	0	0	0	0	112	111
2	000113f07ec002fd	hey man, i'm really trying edit war. guy const...	0	0	0	0	0	0	233	149
3	0001b41b1c6bb37e	can't make real suggestion improvement wondere...	0	0	0	0	0	0	622	397
4	0001d958c54c6e35	you, sir, hero. chance remember page that's on?	0	0	0	0	0	0	67	47



```
In [22]: # Total length removal
         print ('Origian Length', train.length.sum())
         print ('Clean Length', train.clean_length.sum())

         Origian Length 62893130
         Clean Length 43575187
```

```
In [23]: #Getting sense of loud words which are offensive
         from wordcloud import WordCloud
         hams = train['comment_text'][train['malignant']==1]
         spam_cloud = WordCloud(width=600,height=400,background_color='black',max_words=50).generate(' '.join(
         plt.figure(figsize=(10,8),facecolor='k')
         plt.imshow(spam_cloud)
         plt.axis('off')
         plt.tight_layout(pad=0)
         plt.show()
```

iv) Convert text into Vector

```
In [28]: #  Convert text into vectors using TF-IDF
         from sklearn.feature_extraction.text import TfidfVectorizer
         tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
         features = tf_vec.fit_transform(train['comment_text'])
         x = features
```

- Hardware and Software Requirements , Tools Used

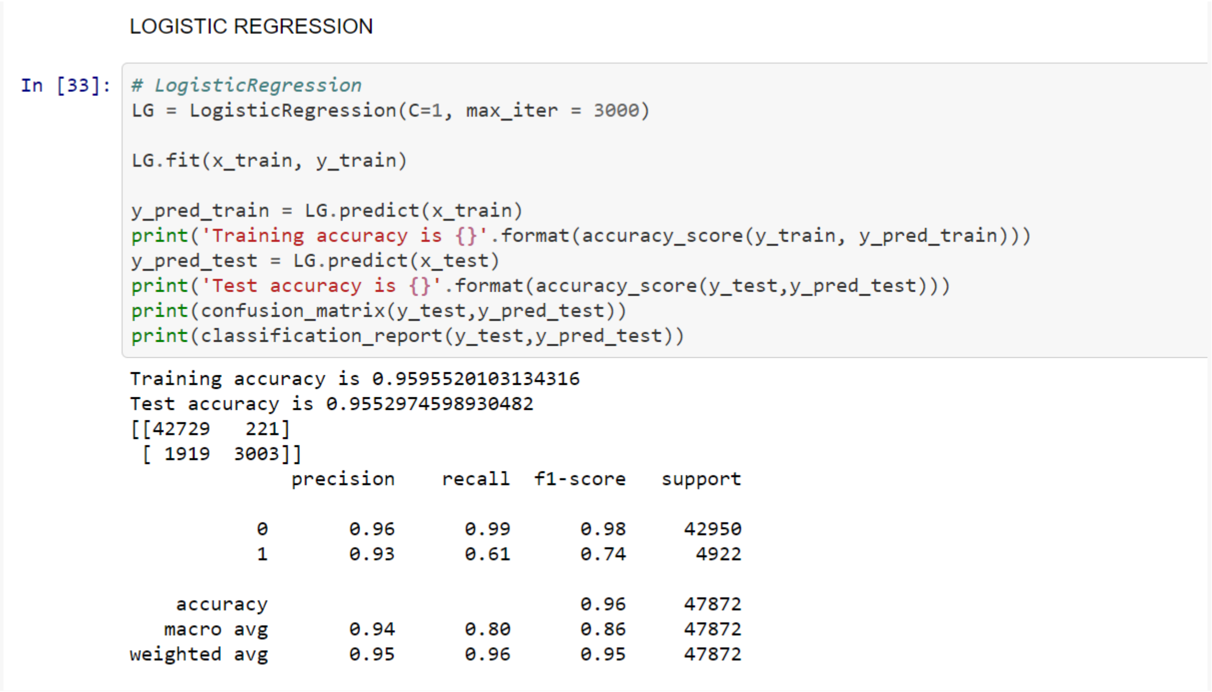
  No Specific requirements except Jupyter Notebook.

Model/s Development and Evaluation

- Identification of possible problem
  This is a Classification problem and we have used Decision tree Classifier, Random Forest Classifier,XGboost,Adaboost ,K-Neighbors classifier to build the model as per the performance the Random forest classifier is chosen to be the best one.

- Run and Evaluate selected models: We ran and evaluated the above mentioned models and chose Random forest classifier as the best model.

- LOGISTICREGRESSION:
  Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression[1] (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1"



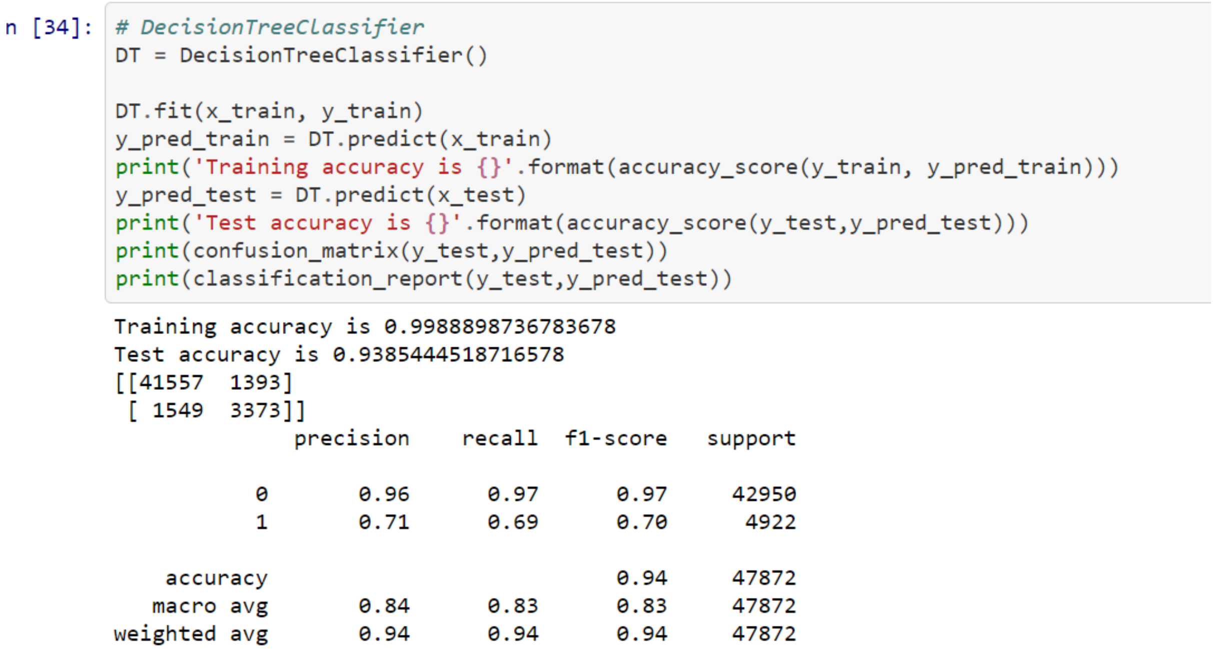
LOGISTIC REGRESSION

In [33]:
```
# LogisticRegression
LG = LogisticRegression(C=1, max_iter = 3000)

LG.fit(x_train, y_train)

y_pred_train = LG.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = LG.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9595520103134316
Test accuracy is 0.9552974598930482
[[42729   221]
 [ 1919  3003]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.93      0.61      0.74      4922

    accuracy                           0.96     47872
   macro avg       0.94      0.80      0.86     47872
weighted avg       0.95      0.96      0.95     47872
```

DECISION TREE CLASSIFIER: Decision trees use **multiple algorithms to decide to split a node into two or more sub-nodes**. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. ... The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

DECISION TREE CLASSIFIER

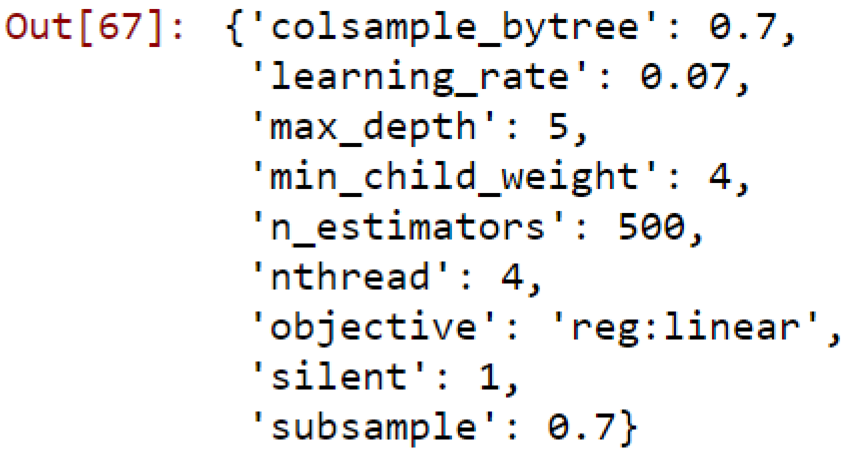


```
n [34]: # DecisionTreeClassifier
        DT = DecisionTreeClassifier()

        DT.fit(x_train, y_train)
        y_pred_train = DT.predict(x_train)
        print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
        y_pred_test = DT.predict(x_test)
        print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
        print(confusion_matrix(y_test,y_pred_test))
        print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9988898736783678
Test accuracy is 0.9385444518716578
[[41557  1393]
 [ 1549  3373]]
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     42950
           1       0.71      0.69      0.70      4922

    accuracy                           0.94     47872
   macro avg       0.84      0.83      0.83     47872
weighted avg       0.94      0.94      0.94     47872
```

RANDOME FOREST CLASSIFIER: Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. The reason for this wonderful effect is that the trees protect each other from their individual errors

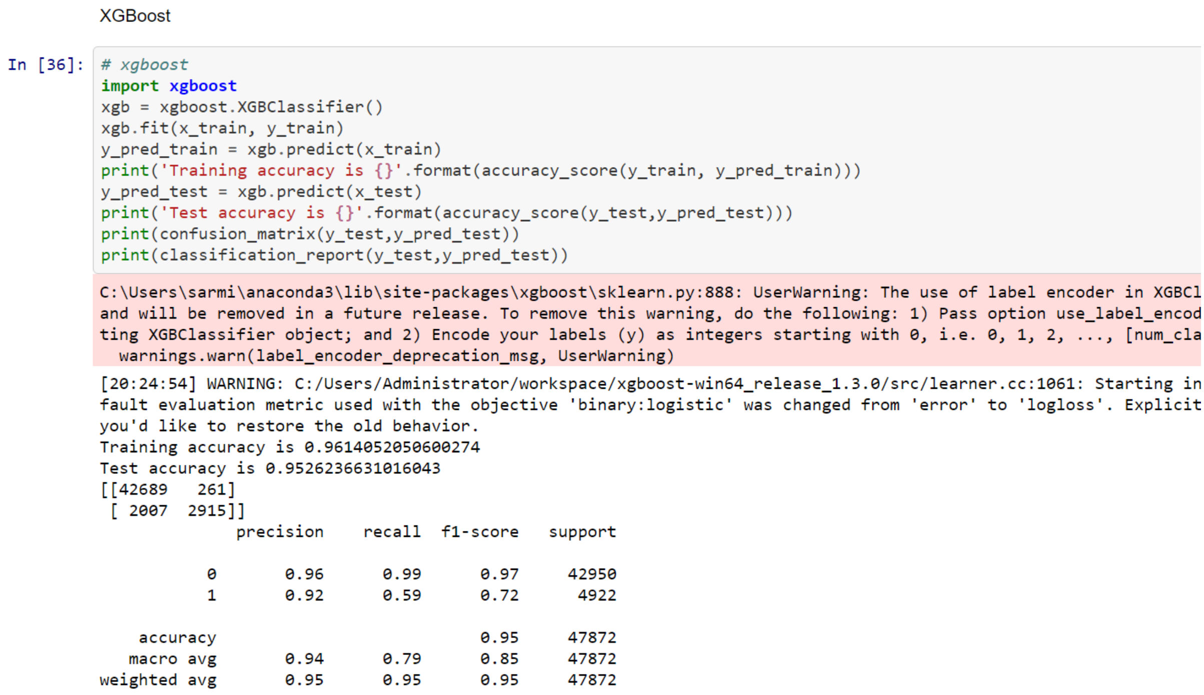
RANDOM FOREST CLASSIFIER



```
Out[67]: {'colsample_bytree': 0.7,
          'learning_rate': 0.07,
          'max_depth': 5,
          'min_child_weight': 4,
          'n_estimators': 500,
          'nthread': 4,
          'objective': 'reg:linear',
          'silent': 1,
          'subsample': 0.7}
```

XGBosst Classifier:

XGBoost is **a decision-tree-based ensemble Machine Learning algorithm** that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) ... A wide range of applications: Can be used to solve regression, classification, ranking, and user-defined prediction problems.

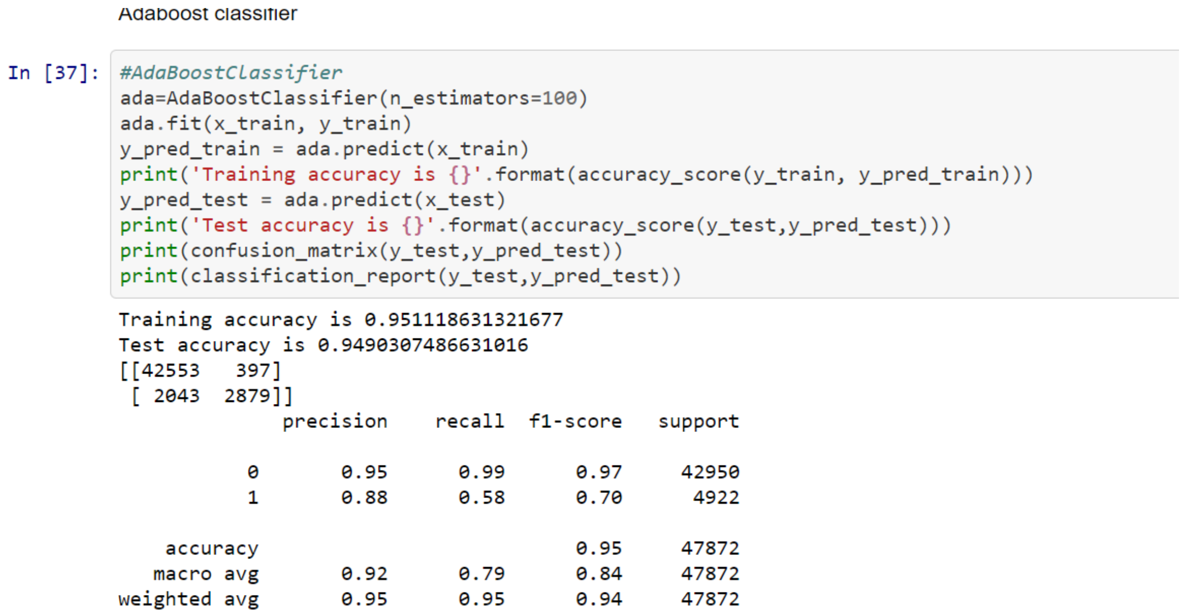


XGBoost

```
In [36]: # xgboost
         import xgboost
         xgb = xgboost.XGBClassifier()
         xgb.fit(x_train, y_train)
         y_pred_train = xgb.predict(x_train)
         print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
         y_pred_test = xgb.predict(x_test)
         print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
         print(confusion_matrix(y_test,y_pred_test))
         print(classification_report(y_test,y_pred_test))
```

```
C:\Users\sarmi\anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBCl
and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encod
ting XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_cla
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[20:24:54] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in
fault evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicit
you'd like to restore the old behavior.
Training accuracy is 0.9614052050600274
Test accuracy is 0.9526236631016043
[[42689   261]
 [ 2007  2915]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     42950
           1       0.92      0.59      0.72      4922

    accuracy                           0.95     47872
   macro avg       0.94      0.79      0.85     47872
weighted avg       0.95      0.95      0.95     47872
```

ADABOOST: An AdaBoost [1] classifier is **a** meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.



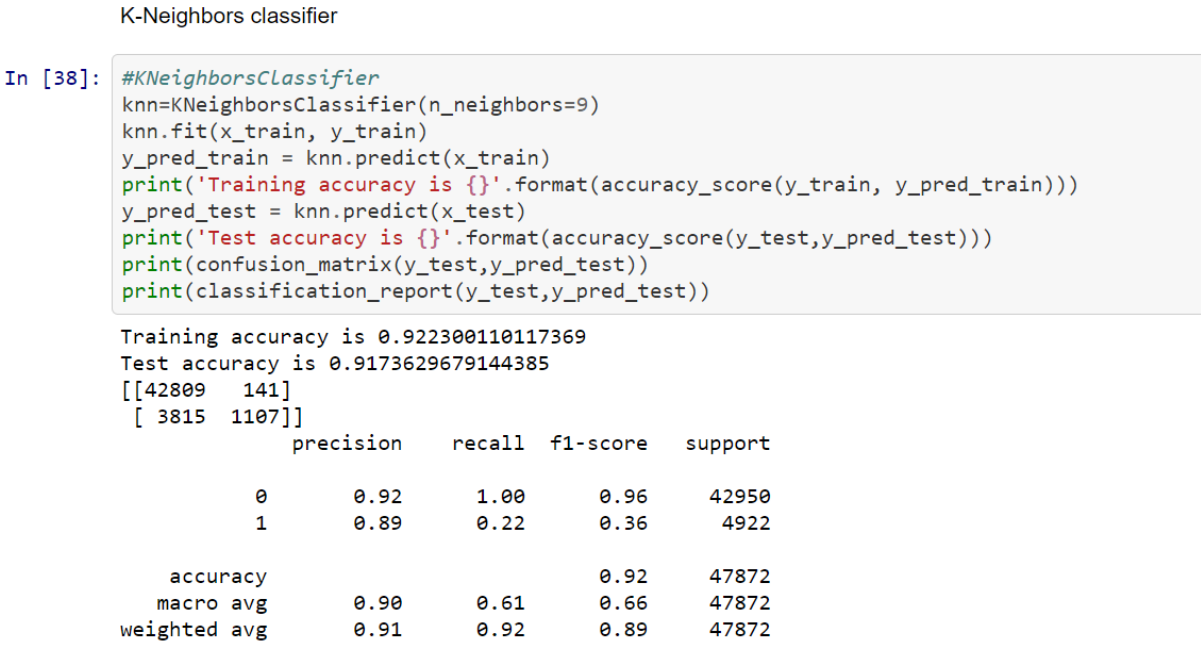
Adaboost classifier

```
In [37]: #AdaBoostClassifier
         ada=AdaBoostClassifier(n_estimators=100)
         ada.fit(x_train, y_train)
         y_pred_train = ada.predict(x_train)
         print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
         y_pred_test = ada.predict(x_test)
         print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
         print(confusion_matrix(y_test,y_pred_test))
         print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.951118631321677
Test accuracy is 0.9490307486631016
[[42553   397]
 [ 2043  2879]]
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     42950
           1       0.88      0.58      0.70      4922

    accuracy                           0.95     47872
   macro avg       0.92      0.79      0.84     47872
weighted avg       0.95      0.95      0.94     47872
```

K-Neighbors classifier:

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.



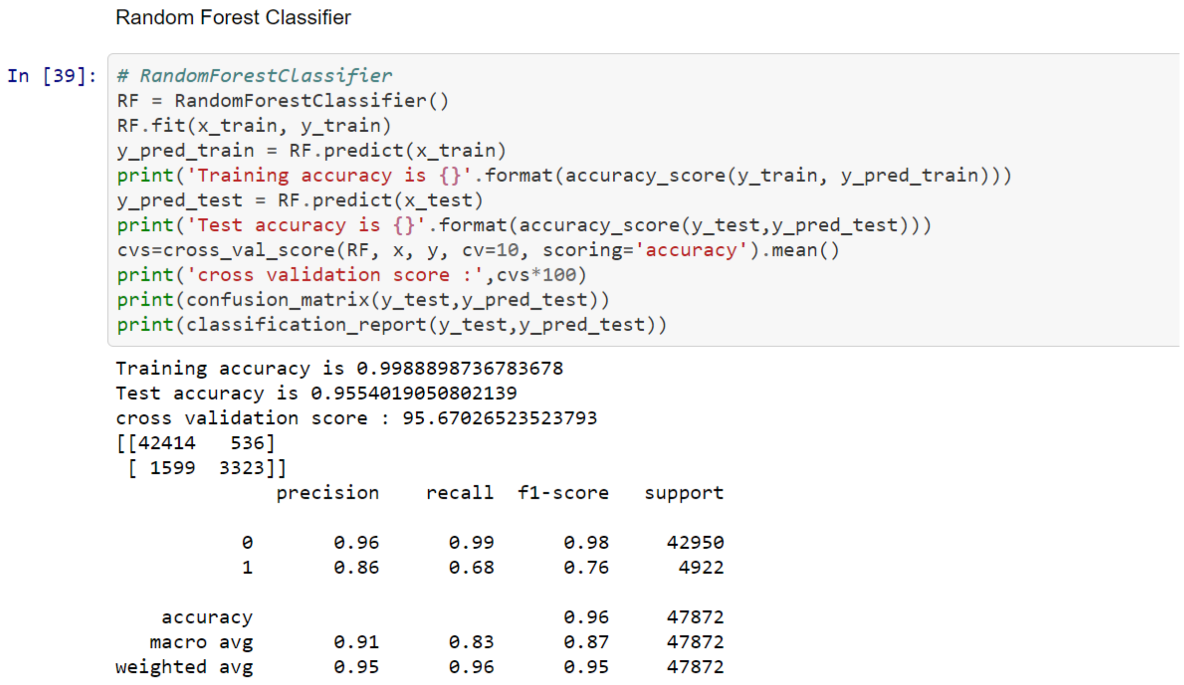
K-Neighbors classifier

```
In [38]: #KNeighborsClassifier
         knn=KNeighborsClassifier(n_neighbors=9)
         knn.fit(x_train, y_train)
         y_pred_train = knn.predict(x_train)
         print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
         y_pred_test = knn.predict(x_test)
         print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
         print(confusion_matrix(y_test,y_pred_test))
         print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.922300110117369
Test accuracy is 0.9173629679144385
[[42809   141]
 [ 3815  1107]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     42950
           1       0.89      0.22      0.36      4922

    accuracy                           0.92     47872
   macro avg       0.90      0.61      0.66     47872
weighted avg       0.91      0.92      0.89     47872
```

- Key Metrics for success in solving problem under consideration

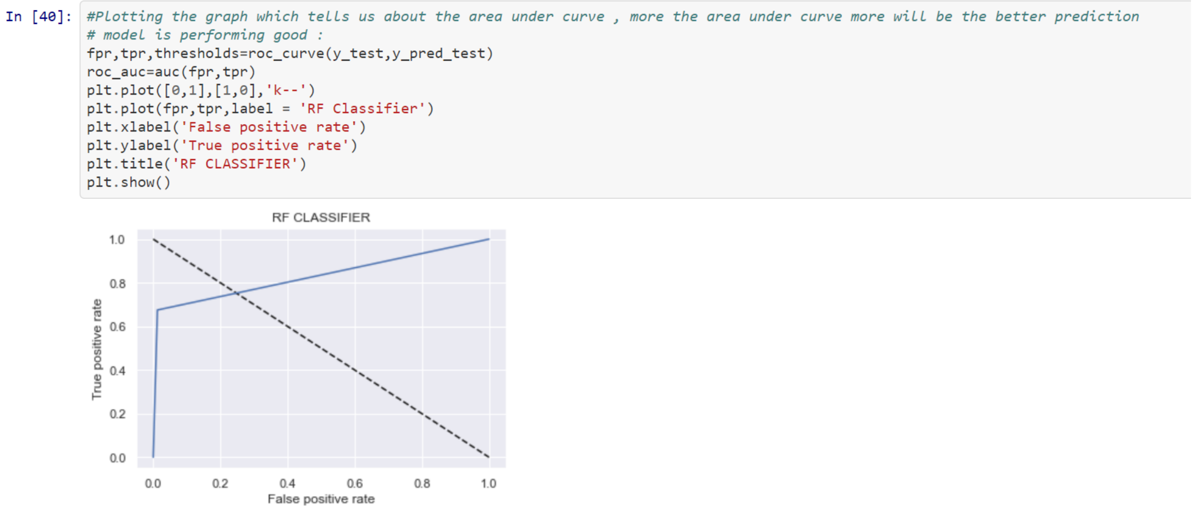
We found the accuracy, confusion matrix and then decided that Random forest is the best model.Tuning the best model further.



Random Forest Classifier

```
In [39]: # RandomForestClassifier
         RF = RandomForestClassifier()
         RF.fit(x_train, y_train)
         y_pred_train = RF.predict(x_train)
         print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
         y_pred_test = RF.predict(x_test)
         print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
         cvs=cross_val_score(RF, x, y, cv=10, scoring='accuracy').mean()
         print('cross validation score :',cvs*100)
         print(confusion_matrix(y_test,y_pred_test))
         print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9988898736783678
Test accuracy is 0.9554019050802139
cross validation score : 95.67026523523793
[[42414   536]
 [ 1599  3323]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.68      0.76      4922

    accuracy                           0.96     47872
   macro avg       0.91      0.83      0.87     47872
weighted avg       0.95      0.96      0.95     47872
```



```
In [40]: #Plotting the graph which tells us about the area under curve , more the area under curve more will be the better prediction
         # model is performing good :
         fpr,tpr,thresholds=roc_curve(y_test,y_pred_test)
         roc_auc=auc(fpr,tpr)
         plt.plot([0,1],[1,0],'k--')
         plt.plot(fpr,tpr,label = 'RF Classifier')
         plt.xlabel('False positive rate')
         plt.ylabel('True positive rate')
         plt.title('RF CLASSIFIER')
         plt.show()
```

CONCLUSION

- Conclusions

According to the performance metrics, Random forest classifier is the best among the others.

• Limitations & Scope for Future

Although we have tried quite several parameters in refining my model, there can exist a better model which gives greater accuracy.

In future, we may be able to add the following to the existing model for enhanced performance

The current project predicts the type or toxicity in the comment.

→ Analyse which age group is being toxic towards a particular group or brand.

→ Add feature to automatically sensitize words which are classified as toxic.

→ Automatically send alerts to the concerned authority if threats are classified as severe.

→ Build a feedback loop to further increase the efficiency of the model.

→ Handle mistakes and short forms of words to get better accuracy of the result